AD-A034 663 ARMY ELECTRONICS COMMAND FORT MONMOUTH N J F/G 17/7
DIGITAL GENERATION OF CONTOUR MAPS FOR RASTER SCAN DISPLAY.(U)
DEC 76 V VAJO
UNCLASSIFIED ECOM-4454 NL

1 OF 1
AD-A
034 663

END
DATE
FILMED
3-5-77
NTIS

AD-A034 663

DIGITAL GENERATION OF CONTOUR MAPS FOR RASTER SCAN DISPLAY

ARMY ELECTRONICS COMMAND, FORT MONMOUTH
NEW JERSEY

DECEMBER 1976

Research and Development Technical Report

ECOM-4454

DIGITAL GENERATION OF CONTOUR MAPS FOR RASTER SCAN DISPLAY

Victor Vajo

Avionics Laboratory

December 1976

ECOM

US ARMY ELECTRONICS COMMAND FORT MONMOUTH, NEW JERSEY 07703

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: ECOM-4454 / 2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): Digital Generation of Contour Maps for Raster Scan Display	5. TYPE OF REPORT & PERIOD COVERED / 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Victor Vajo	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Control Theory Team, Advanced Avionics Systems Tech Area, Avionics Laboratory, Fort Monmouth, NJ 07703	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 1F2 62202 AH85 13.85
11. CONTROLLING OFFICE NAME AND ADDRESS: CG, USAECOM, ATTN: DRSEL-VL-D, Fort Monmouth, NJ 07703	12. REPORT DATE: DECEMBER 1976 / 13. NUMBER OF PAGES: 57
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED / 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer Generated Map
Computers
Computer Mapping Techniques
Television
Map Display
Digital Map Generation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The study proved the feasibility of generating digital contour maps for display on standard TV monitors. Computer programs were written in assembly language for the Singer SKC-2000 Airborne Computer which generate two color (black and white) contour maps for display on a standard 525 line television system.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

NOTICES

TABLE OF CONTENTS

1. INTRODUCTION

This report is concerned with the development of a digitally generated contour map to be displayed on standard raster TV for use in Army aircraft. The requirement for a display of this type is generated by the operations of Army aircraft in nap-of-the-earth (NOE) flight during both day and night operation. NOE flight in this case refers specifically to pilotage at or below tree top level.

Current operations of this type are carried out with the pilot dedicated solely to the task of piloting the vehicle. During this mode of flight, the pilot has the time neither to navigate nor communicate with anyone other than the copilot. For this reason the performance of all other tasks are left to the copilot. In addition, the copilot must cue the pilot as to the character of the upcoming terrain including general navigational instructions. Required terrain information is obtained by the copilot from hand-held maps. The copilot mentally integrates the map contour information and verbally passes upcoming terrain characteristics to the pilot. This task performed by the copilot would be difficult enough under normal conditions, but in the environment of a vibrating helicopter at night, the task becomes even more difficult.

The first phase of this study is directed toward improving the transfer of information concerning the approaching terrain to the pilot, thereby reducing the hazards of NOE night flight.

2. BASIC CONCEPT

Any study designed to solve a particular problem must begin with an attempt to determine whether existing equipment is available to solve the problem being investigated.

Map display systems for aircraft are not a new concept. Currently there are in existance various analog map displays such as map plotters, projected map displays, and also stroke written or CRT type map displays. Each of the above suffers from at least one drawback, namely the fact that they require valuable instrument panel space. In addition, most of the projected map and map plotters present only a north-up display as opposed to an aircraft heading-up display. Those units which can produce a heading-up display, do so at a significant increase in unit cost. Because of these factors and an indication that future Army aircraft may have an integrated digital display and possibly Forward Looking Infra-Red (FLIR) or Low Light Level Television (LLLTV), another avenue of approach was selected for investigation.

An alternative technique to providing a map display is to digitally generate the map (DGM) and display the information in standard raster scan television format. Provided the DGM could be proven feasible, several advantages were immediately apparent. The information could be displayed on panel mounted display (PMD) associated with the FLIR OR LLLTV or even a Helmet Mounted Display (HMD), either alone or superimposed by simply video mixing. The requirement for a separate panel display would thereby be eliminated. In addition, since data for the map would be in digital form, the technique would provide the capability for a much more versatile display. A heading-up display, aircraft position indication, and map scale changes would present no problem using the DGM System.

With the immergence of the microprocessor and the significant reduction in the size and cost of computer memories required for data storage, the DGM appeared to be a reasonable area for investigation.

3. DEFINITION OF DIGITALLY GENERATED MAP SYSTEM

Typical maps used by Army aviators have scales of 1:25,000, 1:50,000 or 1:100,000. Since these maps contain a considerable amount of information, three problems immediately present themselves. The first is how much of the data from the maps must be stored, second, what format or scheme is best for storing the data and third, from where is the digital data to be obtained.

Currently, digital terrain elevation information in grid line format is available from the Defense Mapping Agency for certain areas of the United States. The grid line format refers to the fact that elevation information is stored corresponding to fixed X-Y grid increments over the entire map. Since the size of the map and grid resolution is known, the number of data points can be determined, thus fixing the size of the data base. However, it was felt that data storage in grid format would not offer the best approach, since it requires computer time to generate contour information from the grid data. Therefore, direct storage of data in contour format seemed appropriate.

Contour format means the DGM data will be piece-wise linear approximation to the contour lines. Each pair of consecutive points defines the beginning and end of a line segment. The points are stored in contiguous order. In other words, a line drawn sequentially connecting the points would trace out a contour closing on itself and then begin to trace the next contour interval.

Storing the data in contour format eliminates the computational time required to convert from grid to contour data format. It also provides another distinct advantage in that it facilitates a variable data density. This allows more detailed information to be encoded where terrain variations are severe and less information in areas of less terrain variation. A prime example of the advantage of variable data density (contour format) over fixed data density (grid format) is shown by the fact that grid data provides the same data density over a large lake as it does over the peak of a mountain, whereas the contour format can provide greater data diversity in the mountainous area where it is needed and none over water areas. In using a variable data density, however, the size of the data base required to digitally encode a given size map is undefined. Data base size now becomes a function of the severity of the terrain in the area mapped.

It is possible to develop a computer program to generate a contour data base from the Defense Mapping Agency grid format data base. However, it was felt that development of the data conversion program was too time consuming a project on which to expend much effort prior to the establishment of the feasibility of the DGM system. For this reason, a contour data base was encoded manually from existing maps for usage as a test model. Therefore, in summary, a contour format data base was employed in this study and because of the nature of the data base, the amount of data required for a given map size was variable. Additionally, for the purposes of this study, the data bases were generated manually.

4. DIGITALLY GENERATED MAP EVALUATION SYSTEM

In the laboratory test facility, an airborne digital computer (a Singer SKC-2000) was used to develop the computer programs to generate data for the raster display. A special digital to video converter (DVC) was fabricated in-house specially designed to interface with the airborne computer. The SKC-2000 computer was chosen since an identical computer was installed on the laboratory's CH53 Experimental Vehicle for Avionics Research (EVAR); therefore, after laboratory development of the software programs, the system could be transferred to flight test with minimum difficulty. The design of the DVC was such that it was rugged enough for use during the flight testing.

The SKC-2000 is a 32 bit hexadecimal machine with hardware floating point. Both the laboratory and airborne computers contain sixteen thousand words of memory. Each machine has a teletype with cassettes and a standard size airborne magnetic tape unit. The laboratory model has, in addition, a paper tape punch and reader, a card reader, and a printer/plotter. The printer/plotter allowed the development of the computer programs to be undertaken prior to the fabrication of the DVC. Obviously, the laboratory evaluation system contained sufficient flexibility for a study of this type. All programming was done in assembly language and a listing of the computer assembly language instruction set is given in Appendix A. The instruction set contains several bit manipulation instructions which proved to be extremely useful, especially in encoding the data base.

5. INPUT DATA GENERATION

As previously mentioned, the digital map data (i.e., contour lines) are piece-wise linear approximations to the contour lines. A pair of points define the start and end of each line segment and the sequence of storage of the points gives the path of the contour.

The raw data is obtained in decimal format with X coordinate and Y coordinate specified for each point. Some reference point must be chosen as the origin (0,0) of the data; therefore, the lower left corner of the area to be mapped was selected. A data point beginning a new contour interval is specially flagged to indicate a contour connecting line (non-contour line), enabling the elimination of the line during display.

Since the data in decimal format is not directly usable by the computer, a program was written to convert it to fixed point binary data. The program listing is found in Appendix B. Some use was made of the architecture of the machine in converting to the binary format. Since the computer is capable of both full word (32 bits) and half word (16 bit) addressing, a special scheme for storing the binary data was used. Two types of data words were employed. The first type is called a start point, which is the absolute value of the location of the data point referenced to the origin (0,0). The start point data is encoded as two consecutive 16 bit words, one for X-position and other for Y-position. A start point is identified by the most significant bit of the first 16 bit word being set to one. A hidden line indication is incorporated by using the first bit of the second 16 bit word as that indicator. A one in this bit position signifies the data point is a contour connecting line rather than a contour line itself and should not be displayed. Therefore, 15 bits remain to represent absolute X-position and 15 bits for absolute Y-position. With a scale factor of one (i.e., the least significant bit of the 15 remaining bits represents one foot in ground distance), the maximum possible range of X or Y is $(2^{15} -1)$ or 32,767 feet (approximately 6 miles). This scale factor allows encoding of a map 6 miles by 6 miles. Obviously larger areas may be encoded using larger scale factors with corresponding loss in resolution.

The second type of data word is a delta data word. In order to conserve memory, a delta word format is used to indicate incremental X and Y position referenced to a start word rather than the origin. The delta word is 16 bits in length including both Δx and Δy. The first 8 bits are Δx, the last 8 bits are Δy. The most significant bit of Δx must always be zero indicating it is not a start word. The most significant bit of Δy is used as the hidden line indicator, being set to one if it is a connecting line. There remains 7 bits (6 bits + sign) for indicating Δx and 7 bits (6 bits + sign) for specifying Δy. In other words, with a scale factor of one, changes in X or Y position of less than or equal to $\pm$ $(2^6 -1)$ or $\pm$ 63 feet can be specified using a delta word format.

The delta word format is of significant use in map areas of high information content, since greater detail is encoded by using smaller line segments to describe the contour variations. All of the data bases used in this study were obtained manually and no attempt was made to automate the data base creation.

6. RASTER DISPLAY

Before describing the raster display utilized in this study, a very brief description of standard television systems is given here for comparison.

Standard television uses a 525 line system, meaning that there are 525 horizontal sweep lines or raster lines used in generating a normal 4 by 3 aspect ratio TV picture. The 525 lines are divided into two fields of 262-1/2 raster lines each, called odd and even fields. These two fields are displayed alternately at the rate 1/60 cycle per second. This rate is rapid enough so the eye is not able to perceive any flicker. The camera and associated electronics photographing the scene obviously generates the video in a compatible odd-even field format for transmission. In reference to resolution, the vertical resolution is divided into 525 discrete raster lines, while the horizontal resolution is very nearly continuous.

The raster display used in this study also utilized a 525 line system, but the 4 by 3 aspect ratio picture was not maintained. Rather, it was decided to use a square picture which significantly reduced the complexity of the software. In order to generate the digital equivalent of a TV picture (which is an analog display rather than digital), it was required to form the picture by using a matrix of discrete dots. The study was directed at achieving only a two color level black and white display. No attempt was made to incorporate any shades of gray capabilities. Using only black or white, simplified both computer programming and interface hardware. The dot matrix consisted of an array of 256 by 256 discrete points comprising the picture on the screen, each point capable of having only two possible states either black or white. This binary scheme was compatible with the computer storage of the picture information, thereby requiring a minimum amount of memory. A single bit in the computer contained the information for one pixel on the monitor; if the bit is set to one, it appears as a white dot and, if it is set to zero, it appears as a black dot. As stated before, the TV monitor used was a 525 line system and in order to make the 256 by 256 bit matrix compatible with the 525 line system, the same 256 by 256 matrix was used for both the odd and even fields. In addition, the horizontal sweep was adjusted to obtain a square picture maintaining the same resolution in both the horizontal and vertical direction. Again, this was done to simplify the hardware and reduce computer memory requirements.

7. DIGITAL-TO-VIDEO INTERFACE

The purpose of the digital-to-video interface or digital-to-video converter (DVC) was to accept the computer generated picture information at computer rates and display the information at video rates. To accomplish this, the DVC had incorporated within it two separate banks of 256 by 256 bit memories. As one memory was being written into by the computer, the other was being displayed. When the second memory had been filled by the computer, that memory bank became the display memory and the first became the one being written into by the computer. This ping-ponging of memories continued at the fastest rate allowed by the computer. The switching of memory banks was required, since reading and writing of the same memory simultaneously would cause distortion in the displayed picture. The DVC serially transferred data from the computer by means of direct memory access, eight 32 bit words (256 bits) at a time. This format of transfer was selected since the computer generated one raster line (256 bits) at a time. In this manner, the display was operating asynchronously from the computer, repeating the same digital map until an updated map had been completed. Under all of the operating conditions, the DVC was able to accept data much faster than the computer could generate it.

8. SOFTWARE PROGRAM

The description of the assembly language program developed to generate the digital may be divided into five major sections.

- Determine and save in a temporary memory buffer those data points which are within the field-of-view.

- Rotate the selected data into the aircraft reference system.

- Determine the intersection of the contour lines with the raster lines.

- Store the computed intersections in the output buffer with the proper bit patterns for output to the display.

- Output the computed data.

A more detailed presentation of the program's operation follows.

a. A Determination of Data Points within the Field-of-View. The data base generated in the format described previously is loaded into the computer memory before starting the main program. The next step is to acquire the aircraft parameters required by the program, namely X-position and Y-position relative to the map origin, and aircraft heading. The parameters are generated and varied by means of a control subprogram (listed in Appendix B), which is capable of varying these three parameters as well as the scale factor by fixed increments through the use of control switches on the computer. The entire data base is then scanned to find those data points that are located within a distance R_{max} of the aircraft position relative to the data base. R_{max} is the radius of a circle whose magnitude is equal to one half of the diagonal of the display field-of-view (FOV). The radius of a circle is used rather than a square since the data has not been rotated into the aircraft frame of reference. In order to check the data points with respect to R_{max}, they must be uncompressed. Those points found to be within a distance R_{max} are stored in another area of computer memory in uncompressed format to be used during the intersection process. At this time another function is also performed. During the scanning of the data base some contour lines leave the FOV. At the point where this occurs, a new start word is created. The scan of the data continues until the contour returns to the FOV or the end of the data is reached. If the contour returns to the FOV, a hidden line is created connecting this point and the point at which the contour exited. As stated previously, hidden lines are lines which do not appear on the screen. This operation, in effect, creates a new data base whose boundaries are entirely within the FOV. The creation of the FOV data base reduces the number of points which must be rotated and scanned for possible intersection resulting in reduction of processing time during succeeding operations.

b. Coordinate Rotation of the FOV Data Base. All of the data points in the FOV data base are rotated into the aircraft reference system using the following Euler Coordinate Transformation Equations.

$$X_{ac} = X \cos \psi + Y \sin \psi$$

$$Y_{ac} = Y \cos \psi - X \sin \psi$$

where, ψ is the aircraft heading (X,Y) are the original coordinates (X_{ac}, Y_{ac}) are the coordinates referenced to the aircraft heading.

After the transformation, the FOV data base is in the proper format for the raster line intersection processing.

c. Raster Line Intersection Processing. Raster line processing involves checking all line segments (except hidden lines) defined by the pairs of points in the FOV data base for their possible intersection with a raster line. This operation is performed for each of the 256 horizontal raster lines.

The computation begins with the selection of the raster line located at the uppermost portion of the FOV and continues by successive increments to the raster line located at the bottom of the FOV. Representation of successive raster lines is obtained by incrementing Y $scan_i$ by a fixed Δy scan. In order to compute the intersection of these raster lines with the line segments in the FOV data base, computation of the slope of the segment is necessary.

The remaining computation is a simple determination of the intersection of two straight lines. The determination of the intersections results in a series of X_i's for each Y $scan_i$.

A certain amount of computational time may be saved by first checking to determine if the Y value of a particular raster line lies within the boundaries specified by the Y values of the line segments terminal points. In other words, Y $scan_i$ must lie between Y seg_m and Y seg_{m+1} for an intersection to exist. Slopes are computed only after it has been found that an intersection occurs, thereby resulting in computational savings. The X_i's resulting from actual intersections are stored in their proper bit positions in the output scan word buffer.

d. Formating the Output Scan Word Buffer. The intersection previously determined must next be placed in the proper bit position of the output buffer. Within the computer, a raster line is represented by 256 bits or eight 32 bit scan data words. Each scan data word will contain a one in each bit position for which an intersection occurred. All other bit positions will contain zeros. Using the desired resolution, the determined X_i is divided by the proper scale factor (X scale) to obtain the bit position into which a one must be placed. The scale factor is determined quite simply, namely X scale is the horizontal FOV width divided by 256 bits. The proper bit position is found as follows:

$$\text{Bit position } X_n = \left| X_i / X \text{ scale} \right|$$

This scheme results in only one bit being placed on a raster line for each intersection with that raster line. Actually, this scheme results in a satisfactory representation of contour line segments that are in the range of ± 45 degrees of vertical with respect to the horizontal raster lines. A problem arises as the contour lines begin to approach being parallel to the raster lines. In this case more than one bit must be placed on a raster line in order to make the line look continuous. The obvious extreme case is a contour line exactly parallel to the raster line. If the contour line falls in between two raster lines, a decision is made as to which raster line the contour line should appear on. These various problems are solved by several different bit filling techniques.

The horizontal contour line is the most easily solved problem after determination of the proper raster line has been made. In this case bits are filled on the raster line between the limits determined by the X range of the line segment, namely between X_m and X_{m+1}. Contour lines whose angle with respect to the raster are greater than zero but less than 45 present greater difficulty. This difficulty is increased due to the limitation of having only one raster line in the computer memory at a time rather than having the entire 256 by 256 matrix with which to work. A relatively successful approach was taken to solve the problem. The problem simply stated is -- how many bits must be placed on a raster line for intersections with grazing contour lines. The technique used was to place that number of bits on the raster line that correspond to the absolute value of the reciprocal of the slope of the contour line segment. For example, a line segment intersecting the raster line with a slope of 1/2 (relative to the raster) would have two bits placed on the raster line at the intersection point, a line with a slope of 1/3 would have these bits, those with a slope of 1/4 would have four bits and so forth. Lines with slopes which have fractional parts were rounded to the nearest whole number. This operation significantly increased the computational time.

e. Output to display. The output of the data to the display is facilitated by using the direct memory access (DMA) capability. This requires only that the number of words and the memory location of the first word be specified. The actual transfer is accomplished by one program statement.

9. SUMMARY AND CONCLUSIONS

Results of this study illustrated the feasibility of generating from digital data a dynamic contour map capable of being displayed on a standard raster TV. Figures 1 and 2 show the results as seen on the screen of two different data bases. Figure 1 shows an area one-half mile on each side known as Paw Paw in West Virginia. This data base was obtained manually from a 1:24000 scale map of the area. The contour intervals displayed are at 100 foot intervals as opposed to 20 foot intervals on the source map. There are approximately 4,000 data points in the field-of-view and the time required to generate the picture is about 1 minute and 45 seconds. As can obviously be seen, there is too much information to facilitate easy interpretation. This suggests that this is probably a worst case condition as far as data processing time since any more information would be of no benefit. One point which must be made is that, even though the display is cluttered with information, it does not show all of the information contained on the source map. This suggests that a one-to-one correspondence between paper map and the digital display of a map is not possible within hardware constraints.

Figure 2 shows a simplified data base of the same area as Figure 1. The data base of Figure 2 contains 250 data points and requires only 10 seconds to generate the full picture. Admittedly, this map is rather sketchy and in no way approximates a typical paper map. However, looking a little more closely at the map, the general trend of the terrain is much more obvious in Figure 2 than in Figure 1. The winding river can be seen and the trend of the mountains is more obvious. This is possible with a reduction in the data storage requirement by a factor of 16 and a reduction in computational time by a factor of 10.

Two possible conclusions can be made from the results of Figures 1 and 2. The first is that if a large amount of terrain detail is desired, possibly some format other than standard map contour format should be investigated, since a one-to-one representation of paper maps is not possible. The second conclusion is that if only the trend of the terrain is needed, this technique could be employed directly with satisfactory results.

In reference to the time required to generate the maps, these times may seem to be long; however, no extreme effort was made to achieve rapid execution of the program. It is felt that with minimum effort the program execution times could be reduced significantly. A factor that should be kept in mind is that a helicopter traveling at 20 knots as it does in nap-of-the-earth flight would not necessarily require rapid updates of the map. In fact, if the map could be continuously updated, the continuous movement of the map display would be confusing rather than helpful.

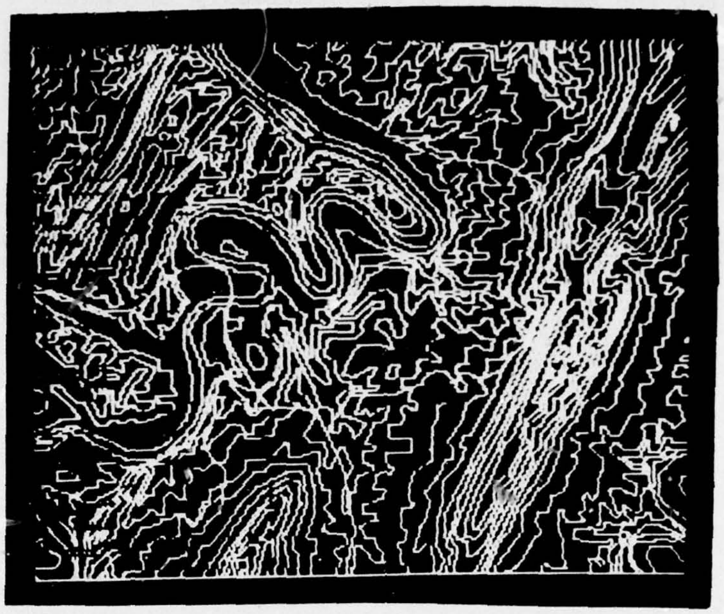

Figure 1. Paw Paw, West Virginia - 4,000 data points

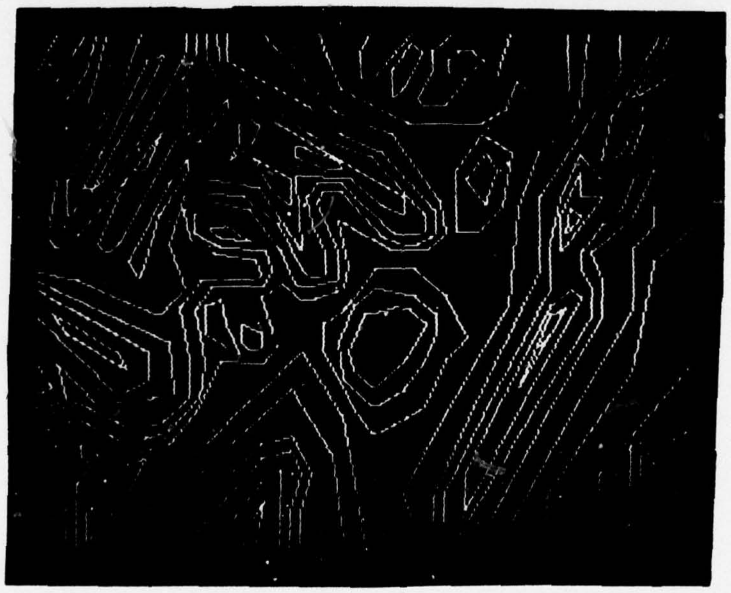

Figure 2. Paw Paw, West Virginia - 250 data points

It is anticipated that further investigations will be made in the area of digital map generation. These investigations will include the generation of a shades of gray or even color map as opposed to the two color black or white display used in this study. Another area of possible investigation will be alternate formats for the map data, as was previously mentioned. Possibilities for alternate formats may include; slope shading, relief shading, and ridge valley lines. Future investigations will also include flight testing of a digital map system and a projected map display.

This study was successful in that it illustrated the capability of developing software and hardware to generate contour maps from digital data for display on raster TV. The study also pointed out some of the shortcomings of the technique and determined possible areas for future investigations.

APPENDIX A

SINGER KEARFOTT COMPUTER NO. 2000 INSTRUCTION SET

OP-CODE	LENGTH	MNEMONIC	OPERATION DESCRIPTION	OPERATION SUMMARY
00010...00.....	S	SLLD	Shift A, B Left Logically	Shift By EA
00010...01.....	S	SLCD	Shift A, B Left Circularly	Shift By EA
00010...10.....	S	SLL	Shift A Left Logically	Shift By EA
00010...11.....	S	SRLD	Shift A, B Right Logically	Shift By EA
00011...00.....	S	SRAD	Shift A, B Right Algebraically	Shift By EA
00011...01.....	S	SRCD	Shift A, B Right Circularly	Shift By EA
00011...10.....	S	SRA	Shift A Right Algebraically	Shift By EA
00011...11.....	S	SRC	Shift A Right Circularly	Shift By EA
000100..........	S	LDA	Load A Register	(EA) → A
000101..........	L	LDA		
000110..........	S	STX	Store Index Register	(XR) → EA
000111..........	L	STX		
001000.........0				
001000.........0	S	ICN	Test XR and Skip On Not Equal	Skip if (XR) $\neq$ (EA)
001001.........0	L	ICN		
001001.........1	L	ICL	Test XR and Skip on Less Than	Skip if (XR) < (EA)
001100..........	S	LAE	Load A With EA	EA → A
001101..........	L	LAE		
001110..........	S	STA	Store A Register	(A) → EA
001111..........	L	STA		
010000000.......	S	NOP	No Operation	No Operation
010000001.......	S	EMI	Enable Memory Interrupts	SR14 Is Set To 1
010000010.......	S	DPI	Disable Program Interrupts	SR15 Is Set To 0
010000011.......	S	DMI	Disable Memory Interrupts	SR14 Is Set To 0
010000100.......	S	EPI	Enable Program Interrupts	SR15 Is Set To 1
010000101.......	S	HLT	Halt	Halts If Test Equipment Signal Is Present.
010000110.......	S	SET	Set Selected Program Flags	Sets Indicated Flags To 1
010000111.......	S	RST	Reset Selected Program Flags	Resets Indicated Flags To 0
010001000.......	S	CFX	Convert Floating To Fixed	(A, B) → A, B
010001001.......	S	CXF	Convert Fixed To Floating	(A, B) → A, B
010001010.......	S	EAB	Exchange A And B	(A) → B, (B) → A
010001011.......	S	SHM	Set Halfword Mode	SR Bit Set To 1
010001100.......	S	RHM	Reset Halfword Mode	SR Bit Reset To 0
010001101.......	S	LXA	Load Index Register From A	(A) → XR (18 Low Order Bits)
010010........0.	S	DOR	Data Output From A Register	
010010........1.	S	DIA	Data Input To A Register	
010011........0.	L	DOS	Data Output From Memory	
010011........1.	L	DIM	Data Input To Memory	
010100..........	S	LDB	Load B Register	(EA) → B
010101..........	L	LDB		
010110..........	S	LDX	Load XR Register	(EA) → XR
010111..........	L	LDX		
01100000........	S	JU, JRU	Jump Unconditional	Jump To EA
0110010000110...	L	JU, LGU	Jump Unconditional	Jump To EA
01100001........	S	JN, JRN	Jump If A $\neq$ 0	Jump To EA if (A) $\neq$ 0
0110010100100...	L	JN, JAN	Jump If A $\neq$ 0	Jump To EA If (A) $\neq$ 0

Abbreviations: ( )	Contents of	CARRY	Carry Statis Bit
A	A Register	MR	Interrupt Mask Register
B	B Register	SR	Statis Register
EA	Effective Address	PC	Program Counter
XR	An Index Register	→	Goes Into
		▲	Floating Point

OP-CODE	LENGTH	MNEMONIC	OPERATION DESCRIPTION	OPERATION SUMMARY
01100010........	S	JG, JRG	Jump If A _ 0	Jump To EA If (A) ≥ 0
0110011000100...	L	JG, JAG	Jump If A ≥ 0	Jump To EA If (A) ≥ 0
01100011.	S	JL, JRL	Jump If A < 0	Jump To EA If (A) < 0
0110011100100...	L	JL, JAL	Jump If A ≤ 0	Jump To EA If (A) ≤ 0
011001...0011...	L	JGW	Jump On Switch	Jump To EA If Switch On
011001... 010...	L	JGS	Jump On Statis Bit	Jump To EA If Statis Bit On
011001....001...	L	JGF	Jump On Program Flag	Jump To EA If Any Flag Tested Is On
011001...0000...	L	JS	Jump To Subroutine	(PC)+2→EA Indirectly, Jump To EA+1
01101..........0	L	IMP	Modify Index Register Positive	(XR)+(EA) → XR
011010.........1	S	IMN	Modify Index Register Negative	(XR)−(EA) → XR
011011.........1	L	IMN		
011100....	S	RTA	Return Address	Jump Indirect Via EA
011101..........	L	RTA		
011110..........	S	STB	Store B Register	(B) → EA
011111..........	L	STB		
100000..........	S	AND	Logical And	(A) AND (EA) → A
100001..........	L	AND		
100010..........	S	SAM	Skip On A Register Masked	Skip Unless (A) and (EA) = 0
100011..........	L	SAM		
100100..........	S	MLF	Multiply - Floating Point	(A) $\overset{\blacktriangle}{*}$ (EA) → A, B
100101	L	MLF		
100110..........	S	AFD	Add Floating Double Precision	(A, B) $\overset{\blacktriangle}{+}$ (EA,EA+2) → A, B
100111..........	L	AFD		
101000..	S	ADU	Add Upper - Fixed Point	(A) + (EA) + Carry → A
101001..........	L	ADU		
101010.	S	ADL	Add Lower - Fixed Point	(B) + (EA) → A
101011..........	L	ADL		
101100.........0	S	DVF	Divide - Floating Point	(A, B) $\overset{\blacktriangle}{/}$ (EA) → A, Remainder → B
101101.........0	L	DVF		
10110..........1	L	STS	Store Status Register	(SR) → (EA)
101110.........0	S	ADF	• Add - Floating Point	(A) $\overset{\blacktriangle}{+}$ (EA) → A
101111.........0	L	ADF		
10111...... ...1	L	LDS	Load Statis Register	(EA) → SR
110000..........	S	LOR	Logical OR	(A) OR (EA) → A
110001..........	L	LOR		
110010..........	S	EXO	Exclusive OR	(A) XOR (EA) → A
110011..........	L	EXO		
110100..........	S	MUL	Multiply - Fixed Point	(A) * (EA) → A, B
110101..........	L	MUL		
110110	S	SFB	Subtract - Floating Double Precision	(A, B) $\overset{\blacktriangle}{-}$ (EA, EA+2) → A, B
110111..........	L	SFB		
111000..........	S	SBU	Subtract Upper - Fixed Point	(A) - (EA) - Carry → A
111001..........	L	SBU		
111010..........	S	SBL	Subtract Lower - Fixed Point	(B) - (EA) → B
111011..........	L	SBL		
111100.........0	S	DVD	Divide - Fixed Point	(A, B)/(EA) → A, Remainder → B
111101.........0	L	DVD		
11110..........1	L	STI	Store Interrupt Mask Register	(MR) → EA
111110.........0	S	SBF	Subtract - Floating Point	(A) $\overset{\blacktriangle}{-}$ (EA) → A
111111.........0	L	SBF		
11111..........1	L	LDI	Load Interrupt Mask Register	(EA) → MR

APPENDIX B

LISTINGS OF MAP GENERATOR COMPUTER PROGRAMS

```
*   CONTROL PROGRAM FOR MAP DISPLAY
*
*
*
*
*********************************************************************
*       INSTRUCTIONS
*   SW5 -- X AIRCRAFT POSITION DRIVE
*   SW4 -- Y AIRCRAFT POSITION DRIVE
*   SW3 -- AIRCRAFT HEADING DRIVE
*   SW2--- SCALING FRESOLUT
*   SW0 -- SIGN OF INCR, ON = -1, OFF = +1
*********************************************************************
COSFA      SETX 0444 STRT LOC OF SUBR COS
SINFA      SETX 06C6 STRT LOC OF SUBR SIN
RDDATA    SETX  8120
FOVPTS    SETX  81D0
DTRINTR   SETX  8320
ST1       SETX  8000
*
          ORG   ST1
FRASTER    DEC    256.000
FRESOLUT   DEC   -10.
FXACPOS    DEC    1280.
FYACPOS    DEC    1150.
HEADING    DEC   0.0
SINSI      HEX    0
COSSI      HEX    7FFFFFFF
FSINSI     HEX   0
FCOSSI     HEX   0
INWORDS   HEX   0    NUMBER OF WORDS INPUT
FOVWDK     HEX    0    NO. OF WORDS IN FOV
TEMP       BSS   4
SUBROUT    BSS   8
*
RADIAN     DEC   57.3
ONE        DEC   1.0
FTWO       DEC   2.0
FFIVE      DEC   5.0
FTEN       DEC   10.0
FHUNDR     DEC   100.0
MINUSONE   DEC   -1.0
CONVRTN    HEX   0
BIT1       HEX   80000000
BIT32      HEX   00000001
PLUSONE    HEX   7FFFFFFF
*
ST2       SETX  8040
```

```
          ORG    ST2
          JS     RDDATA   GO READ DATA ONE TIME ONLY
          BASE   5,FRASTER
          LDX    5,FRASTER,M
LOOP      LDX    5,FRASTER,M
*
*
START      JGW   SIGN,0.  IS DECR SET
           LDA   ONE     NO, INCREMENT
*
SW3RTN     STA   TEMP
SW5        JGW   XAC,5
SW4        JGW   YAC,4
SW3        JGW   HEAD,3
SW2        JGW   SCALE,2
HEAD3      JU    HEAD2
*
RETURN    JS     FOVPTS   GO DETERMINE POINTS IN FOV
          JS     DTRINTR  GO COMPUTE INTERSECTIONS WITH RASTER LINES
          JU     LOOP
*
SIGN       LDA   MINUSONE
           JU    SW3RTN
*
XAC        LDA   FHUNDR
           MLF   TEMP  MULT BY SIGN
           ADF   FXACPOS
           STA   FXACPOS
           JU    SW4
*
YAC        LDA   FHUNDR
           MLF   TEMP
           ADF   FYACPOS
           STA   FYACPOS
           JU    SW3
*
HEAD       LDA   FFIVE
           MLF   TEMP
           ADF   HEADING
           STA   HEADING
           JU    SW2
*
HEAD2      LDA   HEADING
           DVF   RADIAN
           LDX   6,SUBROUT+6,M
           JS    COSFA
           STA   FCOSSI
           JS    CONV
```

```
        STB  COSSI
*
        LDA  HEADING
        DVF  RADIAN
        LDX  6,SUBROUT+6,M
        JS   SINFA
        STA  FSINSI
        JS   CONV
        STB  SINSI
        JU   RETURN
*
CONV    PTR  CONVRTN
        CFX
        SAM  BIT1
        SAM  BIT32
        JU   SHIFT
        LDB  PLUSONE
        RTA  CONVRTN
SHIFT   SRAD 1
        RTA  CONVRTN
*
SCALE   LDA  FTWO
        MLF  TEMP
        ADF  FRESOLUT
        STA  FRESOLUT
        JU   HEAD3
*
*
        END
        END
>
```

```
*   PROGRAM READS IN INPUT DATA TAPE
*
*
*
*
CHARSV    SETX  8630
TAPERD    SETX  863C
TTWRITE   SETX  8650
*
INWORDS  SETX  8012   NUMBER OF INPUT WORDS
STORE     SETX  8670   BEGINNING LOCATION OF INPUT DATA BASE
*
BEGIN     SETX  8100
          ORG   BEGIN
DARDRTN  HEX   0
SV1       HEX  0
SV2       HEX  0
LNCKRTN   HEX  0
CKOUNT    HEX  0
LKOUNT    HEX  0
EOT       HEX  00000004    END OF TAPE
LSPACE    HEX  20000000    LEADING SPACE
LCRLF     HEX  0D0A0000    LEADING CR LF
RUBS      HEX  0000007F    RUBOUTS
ZIP       HEX  0   ZERO
*
STRTB     SETX  8120
          ORG   STRTB
RDDATA   PTR   DARDRTN
*
* INITIALIZE SECTION
*
          BASE  5,SV1
          LDX   5,SV1,M
*
         LDA   ZIP
         LXA   0         XR0 IS CHAR COUNT
         LXA   2         XR2 IS INPUT WD CNT
         RST  15      RESET ALL FLAGS
          STA  SV1
          STA  SV2
          STA  LKOUNT
          STA  CKOUNT
*
* READ INPUT DATA AND WRITE IT OUT
*
GNC      JS   TAPERD
          LDA  CHARSV
          STA  SV1
          EXO  RUBS IS IT RUBOUT
          JN   NXT
          JU   GNC
NXT       LDA  SV1
          EXO  EOT
          JN   C1
          JGU  START1
C1       JGW   C4,1    PRINT IF SW 1 SET
         JU    C2   OTHERWISE SKIP PRINT
C4        LDB  SV1
          SLLD 24
          JS   TTWRITE
          JS   LINECK
C2        LDA  SV1
          SBU  64,M  HEX  40
          JG   NXT1
          JU   NXT2
NXT1      LDA  SV1
          SBU  71,M    HEX 47 = G
          JG   NXT2
          LDA  SV1
          ADU  9,M  CONV TO HEX FRM ASCII
          STA  SV1
NXT2      LDA  SV1
          AND  15,M
          LOR  SV2   ADD TO REST OF WD
          IMP  0,1,M   INC XR0 BY 1
          ICL  0,4,M   IS XR0 < 4
          JU   C3
          SLL  4
          STA  SV2
          JU   GNC   GET NXT CHAR
*  16 BITS OF INPUT COMPLETED
*
C3        SLL  16  SHIFT TO LEFT HALF WD
          STH  STORE,2  STORE HALF WD
          IMP  2,1,M   INC XR2-- WD COUNT
          LDX   0,ZIP  RESET CHAR CNT
          LDA   ZIP
          STA  SV2
          JU   GNC   GET NXT WD
*
*  LINE CHECKING SECTION
*
LINECK    PTR  LNCKRTN
          LDA   LKOUNT
          ADU  1,M
          STA  LKOUNT
          SBU  32,M
          JG   OTPCRLF
          LDA  CKOUNT  CHAR COUNT
          ADU  1,M     ADD 1
          STA  CKOUNT
          SBU  8,M
          JG   OTPSP
LRTN      RTA  LNCKRTN
OTPCRLF   LDB  LCRLF
          LDA  ZIP
          STA  LKOUNT
          STA  CKOUNT
          JS   TTWRITE
          JU   LRTN
OTPSP     LDB  LSPACE
          LDA  ZIP
          STA  CKOUNT
          JS   TTWRITE
          JU   LRTN
*
*
*
START1    IMN  2,1,M  DECREMENT WD COUNT
          NOP
          STX  2,INWORDS SAVE WORD COUNT
          LDB  LCRLF
          LDA  ZIP
          JS    TTWRITE
         RTA   DARDRTN
*
         END
         END
>
```

```
*    DETERMINE POINTS IN FIELD OF VIEW AND DECOMPRESS DATA
*
*
*
*
*  CALCULATION OF AIRCRAFT PARAMS AND VARBLS
*
*  FLAG ALLOCATIONS
*   FLAG 1--- IMAG FLG
*   FLAG 2--- OUT OF RANGE FLAG
*   FLAG 8--- SAVE OUT OF RANGE DATA
*
*  REGISTER DEFINITIONS
*   XR2 -- INCOMING DATA UP COUNTER
*   XR3 -- STORED DATA COUNTER
*   XR5 -- BASE REGISTER
*   XR7 -- RTM REG
*
BEGIN      SETX  81A0
           ORG   BEGIN
FRASTER    SETX 8000  NUMBER OF RASTER LINES
FRESOLUT   SETX 8002  RASTER RESOLUTION
FXACPOS    SETX 8004  X POSITION
FYACPOS    SETX 8006  Y POSITION
INWORDS   SETX  8012   NUMBER OF INPUT WORDS
FOVWDK    SETX  8014   NUMBER OF WORDS IN FOV
XTRACK     SETX  3E00
YTRACK     SETX  3E02
STORE1     SETX  8670   STARTING LOCATION OF INPUT DATA BASE
STORE2     SETX  A200  STARTING LOCATION OF DATA POINTS IN FOV
FOVRTN    HEX   0
FTWO       DEC   2.0
SQRTTWO    DEC   1.4142
TEMP       BSS   6
XMIN       HEX   0
YMIN       HEX   0
XMAX       HEX   0
YMAX       HEX   0
FXMIN      HEX   0
FYMIN      HEX   0
FXMAX      HEX   0
FYMAX      HEX   0
SIGN       HEX   80000000
BIT9       HEX  00800000
BIT16      HEX   8000
*
STRTC      SETX 81D0
           ORG   STRTC
```

```
FOVPTS    PTR    FOVRTN
*
* CAL MAX AND MIN X AND Y IN FLT PT
*
          BASE   5,FTWO
          LDX    5,FTWO,M
*  DETERMINE FOV RADIUS
          LDA    FRESOLUT
          MLF    FRASTER
          DVF    FTWO    FRASTER*FRESOLUT/2
          MLF    SQRTTWO
          STA    TEMP
*
* FIND LFT EDGE FOV
          LDA    FXACPOS
          SBF    TEMP
          STA    FXMIN
*
* FIND RT EDGE FOV
          LDA    FXACPOS
          ADF    TEMP
          STA    FXMAX
*
* FIND BOTTOM OF FOV
          LDA    FYACPOS
          SBF    TEMP
          STA    FYMIN
*
* FIND TOP OF FOV
          LDA    FYACPOS
          ADF    TEMP
          STA    FYMAX
*
* CONVERT FLT PT TO FIX PT HALF WORD
*
          LDX    1,0,M    RS XR0
LOOP1     LDA    FXMIN,1
          LDB    0,M
          CFX             CONV FLT TO FIXED PT
          SLL    16       CONV TO HALF WD
          STAH   XMIN,1
          IMP    1,2,M
          ICL    1,8,M
          JU     NX1
          JU     LOOP1
*  INITIALIZE SECTION
NX1       LDX    2,0,M
          LDX    3,0,M
```

```
            LDX    7,0,M
* SET STATUS REG FOR FAST SCRATCH PAD OPER.
            LDS    8192,M  SET BIT 6 IN STS REG
            LDA    0,M
            STA    XTRACK
            STA    YTRACK
            RST    15       RESET ALL FLAGS
*
NX2         LDAH   STORE1,2
            SAM    SIGN      SKP IF SIGN SET
            JU     NORMENT   GO TO INCR FORMAT
*  START WORD FORMAT
STRTWORD    EXO    SIGN    RS START WD IND
            STAH   XTRACK   SAVE X VALUE
            IMP    2,1,M    INC XR2
            LDAH   STORE1,2  GET Y     .
            SAM    SIGN    IS IMAG BIT SET
            JU     NX3     NO
            SET    1       YES, SET IMAG FLAG
            EXO    SIGN    RESET IMAG BIT
NX3         STAH   YTRACK    SAVE Y VALUE
            JU     COMP1
*
*  COMPRESSED WORD FORMAT
*
NORMENT      SAM    BIT9   IS IMAG BIT SET
            JU     NX4
            SET    1   SET FLG 1 -- IMAG FLG
NX4         SLL    1   SH SIGN X TO SGN A REG
            SRA    9   SIGN EXTEND TO 16 BITS
            ADUHR XTRACK,7   ADD HF WD TO FSP
*  WORK ON Y VALUE
            SLL    17  SGN Y TO SGN A REG
            SRA    9   SIGN EXTEND
            ADUHR YTRACK,7  ADD H TO FSP
            JU     COMP1
*
*  CHECK FOR DATA IN FIELD OF VIEW (FOV)
*
COMP1       LDAH   XMAX
            SBUH   XTRACK
            JL     OOR     JMP IF XTRACK>XMAX
            LDAH   XTRACK
            SBUH   XMIN
            JL     OOR     JMP IF XTRACK<XMIN
            LDAH   YMAX
            SBUH   YTRACK
            JL     OOR     JMP IF YTRACK>YMAX
            LDAH   YTRACK
            SBUH   YMIN
            JL     OOR     JMP IF YTRACK<YMIN
```

```
* DATA IS IN FOV
*
          JGF   NX5,2   CK OUT OR RANGE FLG
TABLE     LDAH  XTRACK
          LDBH  YTRACK
          JGF   NX6,1
RETN1     SRA  16
          SLLD  16
          JGF  NX8,8  IS SPECIAL FLG SET
          STA   STORE2,3
RETN2     ICL  2,INWORDS IS IT LAST WORD
          JU    ENDC     YES, END SECT
          IMP   2,1,M    INCR WD CNT
          IMP   3,2,M    INCR OUT WD CNT
          JU    NX2      RECYCLE
*
NX8       STA  TEMP+2
          ICL  2,INWORDS
          JU   ENDC
          IMP  2,1,M
          JU   NX2
*
*   DATA HAS RETURNED TO FIELD OF VIEW
NX5       JGF   NX5A,8  IS ANYTHING STORED
          RST   2
          JU    TABLE
NX5A      LDA   TEMP+2
          STA   STORE2,3
          IMP   3,2,M
          RST   8
          RST   2   RS OUT OF RNGE FLG
          JU    TABLE
*
*  RESTORE IMAG BIT
NX6       EAB
          LOR   SIGN   SET IMAG BIT
          EAB
          RST   1   RS IMAG FLG
          JU    RETN1
*
*  OUT OF RANGE SECTION
*
OOR       JGF   NX7,2
          SET   2       SET FLG IF NOT SET
          RST   8
          JU    TABLE
NX7       SET   8  SET SAVE OOV DATA
          SET   1  SET IMAG BIT
          JU    TABLE
*
ENDC      STX   3,FOVWDK  SAVE NO. WDS
         RTA   FOVRTN
*
         END
         END
```

```
*     ROTATES DATA AND COMPUTES INTERSECTIONS
*
*
*
*
BEGIN       SETX  8290
            ORG   BEGIN
*   DEFINITIONS
*
FRASTER    SETX 8000  NUMBER OF RASTER LINES
FRESOLUT   SETX 8002  RASTER RESOLUTION
FXACPOS    SETX 8004  X POSITION
FYACPOS    SETX 8006  Y POSITION
SINSI     SETX  800A
COSSI     SETX  800C
FSINSI    SETX  800E
FCOSSI    SETX  8010
FOVWDK    SETX  8014   NUMBER OF WORDS IN FOV
TOFORM    SETX   859A   TOP OF FORM ENTRY POINT
LINEFEED   SETX  8590
LINEPRT    SETX  85A8
TVOUT     SETX   7320   TV RASTER OUTPUT ENTRY POINT
STORE2     SETX  A200  STARTING LOCATION OF DATA POINTS IN FOV
SCANWD     SETX  BFE0
ISECTRTN HEX   0
SGNXRTN    HEX  0
RNDRTN     HEX   0
SLPRTN     HEX   0
FONE       DEC   1.0
TWO        DEC   2.0
BIT16      HEX   8000
THIRTEEN   DEC   13.0   HALF THE NUMBER OF RASTER LINES MISSING
IRASTN      DEC   229   NUMBER OF VERTICAL RASTER LINES
SCWDMX     HEX   E   THIS CORESPONDS TO 8 WORDS 256 BITS
HTEN       DEC16 10
BLK1       DEC16  0
PC           HEX   0
YSCANST      HEX   0
YSCAN      HEX   0
YSCANP     HEX   0
YSCANNG    HEX   0
XP         HEX   0
YP         HEX   0
XP2        HEX   0
YP2        HEX   0
X0         HEX   0
SLOPE      HEX   0
RECIPRO    HEX  0   RECIPROCAL OF SLOPE
```

```
XZERO      HEX    0
XLIMIT     HEX    0
XACPOS     HEX   0
YACPOS     HEX   0
TEMP         BSS    6
*
RTHALF     HEX   FFFF0000
ALLONES    HEX   FFFFFFFF    COMPLEMENT MASK
BIT2       HEX   40000000
MASK1        HEX    80000000
MSKIMAG     HEX    7FFFFFFF
*
MASK   HEX  80000000
       HEX  40000000
       HEX  20000000
       HEX  10000000
       HEX  8000000
       HEX  4000000
       HEX  2000000
       HEX  1000000
       HEX  800000
       HEX  400000
       HEX  200000
       ~~HEX~~  ~~100000~~
       HEX  80000
       HEX  40000
       HEX  20000
       HEX  10000
       HEX  8000
       HEX  4000
       HEX  2000
       HEX  1000
       HEX  800
       HEX  400
       HEX  200
       HEX  100
       HEX  80
       HEX  40
       HEX  20
       HEX  10
       HEX  8
       HEX   4
       HEX   2
       HEX   1
ZIP    HEX   0
STRTD      SETX   8320
           ORG    STRTD
DTRINTR  PTR    ISECTRTN
```

```
*
          BASE   5,FONE
          LDX    5,FONE,M
*
*   INITIALIZE
          LDA    ZIP
          LXA    1
          LXA    3
          LXA    4    WORD COUNTER
          RST    4
          RST    1
*   VERTICAL RESOLUTION
*   CONVERT FLOATING RESOLUTION TO INTEGER
          LDA    FRESOLUT
          CFX
          SLL    16
          STAH   HTEN
*
*  TRANSLATION OF DATA POINTS
*
*   CONVERT A/C POSITION TO INTEGER
          LDA    FXACPOS
          CFX
          SLL    16
          STAH   XACPOS
          LDA    FYACPOS
          CFX
          SLL    16
          STAH   YACPOS
*
*   COORDINATE ROTATION OF DATA POINTS
NXLP      LDAH   STORE2,4  LD X PT
          SBUH   XACPOS   SUBTRACT A/C POS
          STAH   XP
          LDAH   STORE2+1,4   LD Y PT
          SAM    MASK1   IS IMAG BIT SET
          JU     NXLP1   NO, GET NXT PT
          AND    MSKIMAG   YES, ELIM IMAG BIT
          SET    8    SET IMAG FLG
NXLP1     SBUH   YACPOS   SUBT A/C POS
          STAH   YP
          LDA    XP
          MUL    COSSI     * COS
          JS     RNDUP
          STA    TEMP     XCOS
          LDA    YP
          MUL    SINSI
          JS     RNDUP
          ADU   TEMP   XCOS+YSIN
          AND   MSKIMAG   ELIM SIGN BIT
          STAH  STORE2,4
          LDA   XP
          MUL   SINSI
          JS    RNDUP
          STA   TEMP   XSIN
          LDA   YP
          MUL   COSSI   YCOS
          JS    RNDUP
          SBU   TEMP   YCOS-XSIN
          AND   MSKIMAG   ELIM SIGN BIT
          JGF   NXLP3,8  IF FLG 8 SET RESTORE-IMAG BIT
NXLP2     STAH  STORE2+1,4
          ICL   4,FOVWDK
          JU    P4NX
          IMP   4,2,M
          JU    NXLP  ROTATE NXT PPT
*
RNDUP     PTR   RNDRTN
          SAM   MASK1
          SAM   BIT16
          RTA   RNDRTN
          ADUH  1,M
          AND   RTHALF
          RTA   RNDRTN
*
NXLP3     RST   8
          LOR   MASK1
          JU    NXLP2
*
*   FIELD OF VIEW LOGIC
P4NX      LDX   4,0,M   RS WD PTR
* DETERMINE FIRST SCAN LINE
          LDB   ZIP
          LDA  FRASTER   NO. RASTER LINES
          MLF  FRESOLUT  RES. FT/RAST. LINE
          DVF  TWO       /2
          STA  TEMP   USE IN MAX MIN CAL
          LDB   ZIP   COMPENSATE FOR MISSING 26 LINES ON TV DISPLAY
          LDA   FRESOLUT
          MLF   THIRTEEN
          STA   TEMP+2
          LDA   TEMP
          SBF   TEMP+2
          STA   YSCANST   Y VALU 1ST SCN LN
          LDA   FONE   LD 1
          STA   PC
```

```
*
*   DETERMINATION OF XMAX & XMIN
*
*   DETERMINE LEFT SIDE LIMIT
          LDA   ZIP
          LDB   ZIP
          SBF   TEMP
          CFX              CONV FLT TO FX
          SLL   16
          STAH  XZERO
*
*   DETERMINE RIGHT SIDE LIMIT
          LDA   TEMP
          CFX             CONV FLT TO FX
          SLL   16
          STAH  XLIMIT
*
*   BEGIN SCAN OF DATA
P4ST      LDA   PC
          SBF   FONE
          MLF  FRESOLUT
          STA   TEMP
          LDA   YSCANST  GT 1ST SCN LIN
          SBF   TEMP
          CFX            CONV FLT TO FIX
          SLLD  16
          STAH  YSCAN   SV CURRENT SCN LN
*
*   DETERMINE HALF OF DEELTA SCAN
          LDB  ZIP
          LDA   FRESOLUT
          CFX
          SLLD  16
          STAH  TEMP   DSC/2
*
          LDAH  YSCAN
          SBUH  TEMP
          STAH  YSCANNG  YSCAN-DSC/2
*
          LDAH  YSCAN
          ADUH  TEMP
          STAH  YSCANP
*
*
*  CHECK FOR IMAG FLAG
*
P4NX0     LDAH  STORE2+3,4   GT YP2
          SAM   MASK1
          JU     P4NX1    NOT IMAG
          JU     P4NX5
*
P4NX1     LDAH   STORE2,4
          JS     SGNX
          STAH   XP
          LDAH   STORE2+1,4
          AND    MSKIMAG   REMOVE IMAG BIT
          JS     SGNX
          STAH   YP
          LDAH   STORE2+2,4
          JS     SGNX
          STAH   XP2
          LDAH   STORE2+3,4
          JS     SGNX
          STAH   YP2
*
*   DETERMINE INTERSECTION
PNX0      LDAH   YP    IS YP LT YSCAN +
          SBUH   YSCANP
          JL     PNX3    YES
*
PNX1      LDAH   YP2    IS YP2 LT YSCAN
          SBUH   YSCAN
          JL     PNX2    YES
          JU     P4NX5   NO, RETURN
*
PNX2      LDAH   YP2    IS YP2 LT YSCAN -
          SBUH   YSCANNG
          JL     COMPINTR    COMPUTE INTERSECTION
*
          SET    1
          LDAH   XP2
          JU     P4NX4A
*
PNX3      LDAH   YP    IS YP LT YSCAN
          SBUH   YSCAN
          JL     PNX6    YES
*
PNX4      LDAH   YP2    IS YP2 LT YSCAN +
          SBUH   YSCANP
          JL     PNX5    YES
          JU     P4NX5    NO, RETURN
*
PNX5      LDAH   YP2    IS YP2 LT YSCAN -
          SBUH   YSCANNG
          JL     PNX9    GO DO DOUBLE FILL
          LDAH   YP2
```

```
         SBUH   YSCAN
         JL     FILL
         JU     P4NX5     NO, RETURN
*
FNX6     LDAH   YP    IS YP LT YSCAN -
         SBUH   YSCANNG
         JL     PNX10     YES
*
PNX7     LDAH   YP2    IS YP2 LT YSCAN -
         SBUH   YSCANNG
         JL     P4NX5     NO, RETURN
*
PNX8     LDAH   YP2    IS YP2 LT YSCAN +
         SBUH   YSCANP
         JL     FILL    GO FILL
PNX9     SET    1 SET DOUBLE FILL FLAG
         LDAH   XP    X0 = XP
         JU     P4NX4A
*
PNX10    LDAH   YP2    IS YP2 LT YSCAN
         SBUH   YSCAN
         JL     P4NX5     YES, RETURN
*
PNX11    LDAH   YP2    IS YP2 LT YSCAN +
         SBUH   YSCANP
         JL     P4NX12
         JU     COMPINTR   COMPUTE INTERSECTION
P4NX12   SET    1
         LDAH   XP2
*
P4NX4A   STAH   X0
         JS     COMPSLP
         JU     SLOPECHEK
*
*   INCREMENT WORD POINTER
*
P4NX5    JGF    DBLFILL,1   IS DOUBLE FILL FLAG SET
         IMP    4,2,M
         RST    4
         ICL    4,FOVWDK
         JU     OUTPUT
         JU     P4NX0
*
DBLFILL  RST    1   RESET DOUBLEFILL FLAG
         JU     COMPINTR   COMPUTE INTERSECTION
*
*  RESTORE NEGATIVE SIGN
SGNX     PTR    SGNXRTN
```

```
          SAM   BI12
          RTA   SGNXRTN
          LOR   MASK1
          RTA   SGNXRTN
*
*  DETERMINATION OF X POSIT. IN SCAN LINE
*   X VALUE ASSUMED TO BE IN A REG
*   XR8 IS WORD COUNT
*   XR1  IS BIT NUMBER
*   FILL FLAG = FLAG 4
*
*  DETERMIN WORD NO. & BIT NO.
*
DTRXPO      STAH  TEMP
          LDX   8,0,M
          LDX   1,0,M
           SBUH  XZERO  SUB LFT DGE LMT
          JL    P4NX10A  OOFOV
          LDB   ZIP
          SRA   15
          DVD   HTEN
          JS    RNDUP
P4NX10    SBUH  32,M
          JL    P4NX11   J LT 0
          ICL   8,SCWDMX
          JU    P4NX5    OOFOV
          IMP   8,2,M    = 0
          JU    P4NX10
*
* LINE ONLY PARTIALLY OUT OF VIEW
P4NX10A   JGF   P4NX10B,4  IS FILL FLG SET
          JU    P4NX5   NO, RETURN
P4NX10B   SRA   15    SCALE FOR DIVISION
          LDB   ZIP
          DVD   HTEN
          JS    RNDUP
          SRA   16  SHIFT TO LOAD XR1
P4NX10C   JN    P4NX10D   JMP IF NEG
          LXA   1        LD XR1
          JU   P4NX13   GO FILL
P4NX10D   ADU   1,M  INCR A REG
          IMN   2,1,M  DECR FILL COUNT
          JU    P4NX10C  IF NOT ZERO GO HERE
          RST   4
          JU    P4NX5  RETURN
*
*
P4NX11    ADUH  32,M
```

```
          SRA    15   SHFT & MLY X 2
          LXA    1
*
*  PLACE BIT IN SCAN WORD
*
P4NX13    LDA    SCANWD,8    GET SCAN WD
          LOR    MASK,1   ADD MASK
          STA    SCANWD,8    RTN WORD
*
          JGF    P4NX14,4    IS FILL FLG SET
          JU     P4NX5     NO, RETURN
P4NX14    RST    4      RESET FILL FLAG
P4NX14A   IMN    2,1,M  DECR FILL COUNT
          JU     P4NX15
          JU     P4NX5   RETURN
P4NX15    IMP    1,2,M
          ICL    1,64,M    IS IT 64
          JU     P4NX16  YES
          LOR    MASK,1
          STA    SCANWD,8
          JU     P4NX14A
*
P4NX16    ICL    8,SCWDMX  IS IT END OF SCAN WDS
          JU     P4NX5   YES, RETURN
          IMP    8,2,M   NO, INCR WD PTR
          LDX    1,0,M
          LDA    SCANWD,8
          LOR    MASK,1
          STA    SCANWD,8
          JU     P4NX14A
*
*   COMPUTE SLOPE
*
COMPSLP   PTR  SLPRTN
          LDAH XP2
          SBU    XP
          JN     COMPN1
          LDAH   XP
          JU     DTRXPO
COMPN1    STA    TEMP
          LDAH   YP2
          SBU    YP
          SRA    15
          LDB    ZIP
          DVD    TEMP
          STA    SLOPE
          RTA    SLPRTN
*
```

```
*    COMPUTE INTERSECTION
*
COMPINTR   JS    COMPSLP
           LDAH  YSCAN
           SBU   YP
           SRA   15
           LDB   ZIP
           DVD   SLOPE
           JS    RNDUP
           ADU   XP
           STAH  X0
*
*  FILL MODE FOR SHALLOW ANGLES REL. TO RASTER
SLOPECHEK LDA   2,M   LD ONE
           LDB   ZIP
           DVD   SLOPE   COMPUTE
           JL    COMPLEMENT
CKSL      JS    RNDUP
           SRA   16
CKSL0      STA   RECIPRO
          LXA   2
          ICL   2,2,M
           JN    CKSL1   RETURN IF 0
           JU    RTNCK   NO, RETURN
CKSL1      LDA   TEMP    XP2-XP
           JG    CKSL2
           LDA   XP
           JU    CKSL3
CKSL2      LDA   XP2
CKSL3      SBU   X0
           JL    RTNCK   RETURN IF NEG
           JN    CKSL4   RETURN IF 0
           JU    RTNCK
CKSL4      SRA   15
           LDB   ZIP
           DVD   HTEN
           SRA   16
           STA   TEMP+4
           LDA   RECIPRO
           LXA   2
           SET   4
           ICL   2,TEMP+4
           LDX   2,TEMP+4
           ICL   2,2,M
           IMN   2,1,M
           JU    RTNCK   NOT 0 OR LESS
           RST   4
RTNCK      LDA   X0
```

```
            JU   DTRXPO
*
* REVERSE COMPLEMENT
COMPLEMENT  SBU  1,M
            EXO  ALLONES
              JU   CKSL
*
*   FILL COMPUTATION
*
FILL     SET   4   SET FILL FLAG
            LDAH  XP
            SBUH  XP2
            SRA   15
            LDB   ZIP
            DVD   HTEN
            JS    RNDUP
            JG     FILLNX
            LDAH  XP2
            SBUH  XP
            SRA   15
            LDB   ZIP
            DVD   HTEN
            JS    RNDUP
            SRA    16
            LXA   2
            LDAH  XP
            JU    DTRXPO
FILLNX      SRA    16
            LXA   2
            LDAH  XP2
            JU    DTRXPO
*
*  OUTPUT SECTION
*
OUTPUT    JS     TVOUT
          JGW    OL1,1   IF SWITCH 1 SET OUTPUT TO LINEPRINTER ALSO
*   CLEAR OUTPUT BUFFER
CLRBUFF   LDA    ZIP   CLEAR A REG
          LXA    4
CLR       STA    SCANWD,4
          IMP    4,2,M
          ICL    4,16,M
          JGU    CONTINUE
          JGU    CLR
CONTINUE  LDA    PC
           ADF    FONE
           STA    PC
           ICL    3,IRASTN
```

```
                JGU    OL2
                 IMP    3,1,M
                 LDX    4,0,M
                 JGU    P4ST
*    OUTPUT TO LINEPRINTER
OL1             JS     LINEPRT
                JGU    CLRBUFF   GO CLEAR OUTPUT BUFFER
*
OL2             JGW    OL3,1    IF LINEPRINT SW SET DO TOP OF FORM
                RTA    ISECTRTN
OL3             JS     TOFORM
                RTA    ISECTRTN
*
                 END
                 END
>
```

```
*** VARIAN PRINTER PLOTTER SUBROUTINE
*
*
*
*
DATAOUT    SETX   BFDA   RASTER LINE DATA OUTPUT--CONTAINS 3 BLANK WDS
START      SETX  8570
           ORG   START
RASTMODE   HEX   0BE0
REMOENAB   HEX   0B20
TOPOFORM   HEX   0BB3
SYNCSTEP   HEX   0B23
*
XR2SV      HEX   0
XR3SV      HEX   0
XR4SV      HEX   0
WDCOUNT    HEX   16
TEMP       HEX   0
*
OUTCMND    HEX   00000A00
*
OPRTN      HEX   0
WAITRTN    HEX   0
ZIP        HEX   0
*
VPPRTN     HEX   0
TOFRTN     HEX   0
LFRTN      HEX   0
*
LF         PTR   LFRTN
           JS    ONE
           LDA   SYNCSTEP
           DOA   22,C,K
           RTA   LFRTN
*
*
TFORM      PTR   TOFRTN
           LDA   REMOENAB
           DOA   22,C,K
           JS    ONE
           LDA   TOPOFORM
           DOA   22,C,K
           RTA   TOFRTN
*
           PTR   VPPRTN
          LDA   ZIP
          STA   DATAOUT      CLEAR LEFT BORDER OF LINEPRINTER
          STA   DATAOUT+2
          STA    DATAOUT+4
           STX    2,XR2SV   SAVE XR2
           STX    3,XR3SV   SAVE REG 3
           STX    4,XR4SV   SAVE REG 4
*
BEGIN      LDA    REMOENAB
           DOA    22,C,K
           JS     ONE
           LDA    RASTMODE
           DOA    22,C,K
           RST    15
          LDX    1,2,M    NUMBER OF VERTICAL BITS PER POINT, -1
LP5        LDX    4,ZIP   WORD COUNTER
LP2        LDX    2,7,M   BIT COUNTER
           LDA    ZIP
           LDBH   DATAOUT,4
           SRLD   16   SHIFT DATA TO RT HF B REG
LOOP       SRCD   1    SHIFT BIT B31 TO BIT A0
           SRA    2    SIGN EXTEND A REG 1 BIT
           IMN    2,1,M
           JU     LOOP
           JGF    LP1,1
           STA    TEMP
           LDA    ZIP
           LDX    2,7,M
           SET    1
           JU     LOOP
*  SET UP TO OUTPUT FORMATTED DATA
LP1        EAB
           JS     OUT
           LDB    TEMP
           JS     OUT
           RST    1
*    INCREMENT WORD POINTER
           IMP    4,1,M
           ICL    4,WDCOUNT  IS LAST WORD
           JU     LP3        YES
           JU     LP2        NO
*    REPEAT LINE
LP3        JS     LF
          IMN    1,1,M    DECREMENT VERTICAL BIT COUNT.
           JU     LP5
LP4        RST    15
*  RESTORE INDEX REGISTERS
          LDX     2,XR2SV   RESTORE XR2
           LDX    3,XR3SV   RESTORE REG 3
           LDX    4,XR4SV   RESTORE
*
```

```
            RTA    VPPRTN
*
*  WAIT ROUTINE
ONE         PTR    WAITRTN
TWO         DIA    22,K
            SAM    14,M
            RTA    WAITRTN
            JU     TWO
*
OUT         PTR    OPRTN
            LDX    3,2,M      BITE COUNTER
LP8         JS     ONE
            LDA    ZIP
            SLLD   8
            LOR    OUTCMND
            DOA    22,C,K     OUTPUT DATA
            IMN    3,1,M
            JU     LP8
            RTA    OPRTN
*
*
            END
            END
>
```

```
*     TV RASTER OUTPUT ROUTINE
*
*
*
*
*     THIS PROGRAM OUTPUTS TO A TV SCREEN
*
START     SETX  8570
TVOUT     SETX  85A8
DATAOUT    SETX  BFE0   OUTPUT DATA BUFFER
*
          ORG   START
TVDATA    HEX   FF705FF0  FIRST 10 BITS ARE COMPL OF NO. OF WDS
RASTRTN   HEX   0         LAST 16 BITS ARE START LOC OF DATA SHFTD 1 BIT
ZIP       HEX   0         TO THE RIGHT (BFE0)
XR4SV     HEX   0
*
          ORG   TVOUT
          PTR   RASTRTN
          STX   4,XR4SV   SAVE INDEX REGISTER
          LDA   TVDATA
          DOA   19,K
          EMI
RCK       DIA   19,K
          SAM   1,M   CHECK WORD COUNT ZERO
          JU    RCK
          LDA   ZIP   CLEAR BUFFER
          LXA   4
CLR       STA   DATAOUT,4
          IMP   4,2,M
          ICL   4,16,M
          JU    CONTINUE
          JU    CLR
CONTINUE  LDX   4,XR4SV
          RTA   RASTRTN
*
          END
          END
```

```
* PAPER TAPE READ AND TELETYPE OUTPUT ROUTINE
*
*
*
BEGIN     SETX  8630
          ORG   BEGIN
CHARSV    HEX   0
TRRTN     HEX   0
TTWRRTN   HEX   0
MASK      HEX   80000000
ZIP       HEX   0
ENHSR      HEX   02DA1AED
*
* PAPER TAPE READ ROUTINE
*   PROGRAM READS 7 BIT ASCII CODE INTO LOC. CHARSV
*   PROGRAM SKIPS BLANKS
TAPEREAD PTR   TRRTN
TAPE0     LDA   ENHSR   ENABLE READER
          DOA   16,C,K   OUTPUT ENABLE
TAPE1     DIA   16,C,K   READ STATUS
          SAM 16384,M IS TAPE READER RDY
          JU    TAPE2  YES, GO READ CHAR
          JU    TAPE1   NO, CK AGAIN
TAPE2     DIA   16,K   INPUT CHAR
          AND   127,M   MASK PARITY
          STA   CHARSV   SAVE CHARACTER
          JN    TAPE3   IS CHAR A BLANK (00)
          JU    TAPE0   YES, READ AGN
TAPE3     RTA   TRRTN   RETURN FROM SUBROUTINE
*
*   TELETYPE OUTPUT ROUTINE
*   DATA ASSUMED TO BE IN B REG. LEFT JUSTIFIED
*
TTYWRITE PTR   TTWRRTN
TTYO      DIA   16,C,K   READ STATUS
          SAM  32768,M   IS TTY BUSY
          JU    TTYO1   NO, OUTPUT CHAR
          JU    TTYO   YES, CK AGN
TTYO1     LDA   ZIP   CLEAR A REG.
          SLLD  8   SHIFT 8 BITS TO A REG.
          JN    TTYO3   OUTPUT IF NOT ZERO
          RTA   TTWRRTN   GO TO RTN IF ZERO
TTYO3     LOR   MASK   SET TTY OUTPUT BIT
          DOA   16,K   OUTPUT CHAR.
          JU    TTYO   GO OUTPUT NXT CHAR
          END
          END
>
```

```
*    PROGRAM TO FORMAT AND COMPRESS CONTOUR DAT
*
*
*
*
*
* THIS PROGRAM ACCEPTS DATA IN THE FOLLOWING FOR
*                  *XXXX/YYYY,  .....,XXXX/YYYY,EOF
*    THE * INDICATES HIDDEN LINES
*      EOT IS THE END OF TAPE CHARACTER -- 04
*      EOF IS AN END OF FILE CARD -- MULTIPUNCH 6 8
*
* FAST SCRATCH PAD MEMORY ASSIGNMENTS
SUM         SETX   3E00
SV1         SETX   3E02
*
I14TRP     SETX    7FBC     INTERRUPT 14 TRAP LOCATION
I14RTN     SETX    7FFC     INTERRUPT 14 TRAP RETURN
BEGIN       SETX   90C0
           ORG   BEGIN
INSTR1     JGU    PUNTRP    PUNCH TRAP WAIT LOOP
ENABLP     HEX    02BA616D    ENABLE PUNCH
DISIO      HEX    02DA616D    DISABLE ALL DEVICES
TEMP        BSS    8
INT14      HEX    20000000
RUBOUTS     HEX   7F7F7F7F
EOT         HEX   00000004
EOF        HEX    0000003E    END OF FILE CHARACTER
ASTER       HEX   0000002A
SPACE       HEX   00000020
COMMA       HEX   0000002C
SLASH       HEX   0000002F
SIGN        HEX   80000000
LMINUS     HEX    2D000000
LSPACE     HEX    20000000
MASKZ       HEX   7FFFFFFF
MASK30      HEX   00000030
MASK40      HEX   00000040
DRANGE      HEX   0000003F
SCFACT      HEX   00000001
ONE         HEX   00000001
TEN         HEX   0000000A
HUND        HEX   00000064
THOU        HEX   000003E8
TENTHOU     HEX   00002710
*
DATAWD      HEX   0
XVALUE      HEX   0
YVALUE      HEX   0
```

```
XBINARY    HEX  0
YBINARY    HEX  0
XTOTAL     HEX  0
YTOTAL     HEX  0
XDIFF      HEX  0
YDIFF      HEX  0
MAXDEL     HEX  0
COMPDATA   HEX  0
ASCTRTN    HEX  0
MESSORTN   HEX  0
KOUNT      HEX  0
DUMMY      HEX  0
RTN1       HEX  0
RTN2       HEX  0
PUNRTN    HEX   0
*
          JGW  *+4,0  IS SW 0 SET
          JS   MESSO NO TYPE MESSAGE
*
* PUNCH LEADER
*
LEADER    LDX  1,25,M YES
          LDB  RUBOUTS
          JS   PUNCHR PUNCH LEADER
          IMN  1,1,M
          ICL  1,1,M
          JGU  LEADER+2
* INITIALIZE SECTION
*
INIT      LDX  1,0,M
          LDX  2,0,M
          LDX  3,0,M
          LDX  4,0,M
          LDX  5,0,M
          LDX  6,0,M
          LDX  7,0,M
          RST  15      RESET ALL FLAGS
          SET  1       SET FLAG 1
          LDA   0,M    CLEAR A REGISTER
          STA  DATAWD
*
*
* READ CARD
*
RC        LDX   8,0,M   CLEAR BUFFER POINTER
          JS    RDPRNT   READ A CARD AND LIST ON LINEPRINTER
* CHECK FOR END OF FILE CARD
          LDA   0,M
          LDBH    CARDBUFF    LOAD FIRST CHARACTER ON CARD
          SLLD    8      SHIFT CHARACTER TO A REGISTER
          LXA     9     LOAD INDEX REGISTER FOR CHECK
          ICN     9,EOF IS IT AN END OF FILE
          JGU     PNCHEOF     GO PUNCH AN EOF ON PAPER TAPE
* BEGIN CHECK OF CHARACTERS
CK0       LDA     0,M    CLEAR A REGISTER
          LDBH    CARDBUFF,8    GET CHARACTER
          SLLD    8      SHIFT CHARACTER TO A REGISTER
          STA TEMP    SAVE CHARACTER
          LXA     9     TRANSFER TO REGISTER FOR CHECKING
          ICN     9,SPACE    IS IT A SPACE
          JGU     INCR  NO, GO INCREMENT BUFFER POINTER
* CHECK IF IT IS A NUMBER
          ICL     9,58,M  IS IT LESS THAN 58
          JGU     CK1    NO, NOT A NUMBER
          ICL     9,48,M  YES, IS IT LESS THAN 48
          JGU     PACK  NO, IT IS A NUMBER - GO PACK
CK1       ICN     9,SLASH    IS IT A SLASH
          JGU     XEND    YES, TERMINATE X FIELD
          ICN     9,COMMA  IS IT A COMMA
          JGU     YEND    YES, TERMINATE Y FIELD
          ICN     9,ASTER  IS IT AN ASTERISK
          JGU     HFLAG    YES, GO SET HIDDEN LINE FLAG
          JGU     ERMES1    NO, THEN IT IS AN INVALID CHARACTER
HFLAG     SET     2       YES, SET HIDDEN LINE FLAG
          JGU     INCR        GO INCREMENT BUFFER POINTER
*
* INCREMENT BUFFER POINTER
INCR      IMP     8,1,M    INCREMENT BUFFER POINTER
          ICL     8,80,M    IS IT END OF CARD
          JGU     INCR1    YES, GO TERMINATE LINEPRINTER LINE
          JGU     CK0    NO, GET NEXT CHARACTER
INCR1     LDB     LEFTCR    TERMINATE LINEPRINTER LINE
          JS      LPMESG    OUTPUT CR
          JGU     RC          GET NEXT CARD
*
PNCHEOF   LDB     EOT    LOAD END OF TAPE CHARACTER FOR OUTPUT TO TAPE
          SLLD    24    POSITION CHARACTER FOR OUTPUT
          JS      PUNCHR    GO PUNCH EOT CHARACTER
          HLT
          JGU     LEADER    GO BEGIN AGAIN
*
* NUMBER PACK ROUTINE
*
PACK      LDA    TEMP
          AND    15,M
          STA    TEMP
```

```
         LDA  DATAWD
         SLL  4
         LOR  TEMP
         STA  DATAWD
         IMP  2,1,M
         ICL  2,5,M
         JGU  ERMES2  INPUT DATA OUT OF RANGE
         JGU   INCR      GO INCREMENT BUFFER POINTER
*
* TERMINATE FIELD NO. 1
*
XEND    LDA  DATAWD
         STA  XVALUE
         LDA  0,M
         STA  DATAWD
         LDX  2,0,M
         JGU   INCR      GO INCREMENT BUFFER POINTER
*
* TERMINATE FIELD NO. 2
*
YEND     LDA  DATAWD
         STA  YVALUE
         LDA  0,M
         STA  DATAWD
         LDX  2,0,M
         LDA  XVALUE
         JS   DBC
         STA  XBINARY
         LDA  YVALUE
         JS   DBC     DECIMAL TO BINARY CONV
         STA  YBINARY
         JGU  WDCOMR
*
* DECIMAL TO BINARY CONVERSION ROUTINE
*
DBC       PTR  RTN1
         STA  SV1
         LDS  8192,M  SET SR6
         LDA  0,M
         STA  SUM
         LDB  0,M
         LDX  5,0,M SET UP PTR
         LDA  SV1
SHCYL    AND  15,M  MASK
         MUL  ONE,5  MUL BY UNITS
         SRAD 1
         EAB
         ADUR SUM,7 ADD TO FAST SCR PAD
```

```
          IMP  5,2,M  INC BY 2
          ICN  5,10,M  HAS IT BEEN 5 CHAR
         JGU  SREST  YES
         LDA  SV1    NO
         SRA  4
         STA  SV1
          ICN  5,8,M  HAVE 4 CHAR BEEN PROCESSED
         JGU  *+4
          JGU  SHCYL
         LXA  0
          ICL  0,7,M  XR0 < 7
          JGU  *+4
          JGU  SHCYL
          JGU  ERMES3  CONV DATA OUT OF RANGE
SREST     LDA  SUM
          RTA  RTN1
*
* WORD COMPRESSION ROUTINE
*
WDCOMR   JGF  NX2,1
          JGU  NX1
NX2      LDA  XBINARY
          STA  XTOTAL
          LDA  DRANGE
          MUL  SCFACT
          EAB
          SRA  1
          STA  MAXDEL
          LDA  YBINARY
          STA  YTOTAL
          RST  1      RSET INITIAL FLG
          JGU  STRTWD
NX1       LDA  XBINARY  CK MAG X
          SBU  XTOTAL   FIND DELTA X
          STA  XDIFF  SAVE DIF
          JAG  *+8   JMP IF DELX + OR 0
          ADU  MAXDEL  ADD MAX DELTA
          JAL  STRTWD  JMP IF -
          JGU  *+6
          SBU  MAXDEL
          JAG  STRTWD
          LDA  YBINARY
          SBU  YTOTAL
          STA  YDIFF
          JAG  *+8
          ADU  MAXDEL
          JAL  STRTWD
          JGU  *+6     USE DELTA FORMAT
```

```
        SBU  MAXDEL
         JAG  STRTWD
*
* DELTA WORD FORMATTING ROUTINE
*
         LDA  XBINARY
         STA  XTOTAL
         LDA  YBINARY
         STA  YTOTAL
         IMP  1,1,M
         LDA  0,M
         STA  COMPDATA
         LDA  XDIFF
         AND  127,M  MASK ANY GARBAGE
         LOR  COMPDATA
         SLL  8
         STA  COMPDATA
         LDA  YDIFF
         AND  127,M
         JGF  NX3,2  IS IMAG FLG SET
         JGU  *+4
NX3      LOR  128,M
         LOR  COMPDATA
         SLL  16
         EAB
         RST  2     RESET IMAG FLG
         JS  ASCNUM
        LDB   LSPACE   LOAD SPACE
        JS    LPMESG   OUTPUT TO LINEPRINTER
         JGU  CONTINUE
*
*
*  OUTPUT TO PAPER TAPE AND
* OUTPUT TO TTY IN ASCII NUMBERS
*      ASSUMES DATA IN B REGISTER
* 16 BIT OR 4 CHAR OUTPUT IN HEX
*
ASCNUM   PTR  ASCTRTN
         LDX  5,0,M
         LDA  0,M
         STA  TEMP+2   CLR
ASCYL    SLLD  4       SHIFT CHAR TO A
         STA  TEMP   SV CHAR
         SBU  9,M    SUBTRACT 9
         JAL  THIRTY  JMP IF < 9
         JAN  FORTY   JMP IF NOT 0
         JGU  THIRTY
FORTY    LOR  MASK40  ASCII LETTER CODE
```

```
         JGU  *+6
THIRTY   LDA  TEMP  RECOVER CHAR
         LOR  MASK30  ASCII NUMBER CODE
         STA  TEMP   SV CHAR CODE
         LDA  TEMP+2  LOAD CURRENT WD
         SLL  8    MK RM FO NX CHAR
         LOR  TEMP   ADD NX CHAR
         STA  TEMP+2 SV CURRENT WD
         LDA  0,M
         IMP  5,1,M
         ICL  5,4,M
         JGU  *+4
         JGU  ASCYL
         LDB  TEMP+2
        JS    LPMESG   OUTPUT TO LINEPRINTER
         LDB  TEMP+2
         JS   PUNCHR
         RTA  ASCTRTN
*
* START WORD FORMATTING ROUTINE
*
STRTWD   LDA  XBINARY
         STA  XTOTAL
         SLL  16
         LOR  SIGN    SET SIGN BIT
         IMP  1,1,M   INCR WORD COUNT
         EAB
         JS   ASCNUM   OUTPUT TO TTY
         LDB  LMINUS    LOAD DASH
        JS    LPMESG   OUTPUT TO LINEPRINTER
         LDA  YBINARY
         STA  YTOTAL
         SLL  16
         JGF  PL6,2  CK IMAG FLG
         AND  MASKZ
         JGU  *+4
PL6      LOR  SIGN
         RST  2
         IMP  1,1,M
         EAB
         JS   ASCNUM   OUTPUT Y TO TTY
        LDB   LSPACE   LOAD SPACE
        JS    LPMESG   OUTPUT TO LINEPRINTER
         JGU  CONTINUE
*
CONTINUE LDA  0,M   CLEAR A REGISTER
        STA   DATAWD
        JGU   INCR      GO INCREMENT BUFFER POINTER
```

```
*
*       ERROR MESSAGES
*
ERMES1    LDA  5,M
          STA  KOUNT
          LDB  TEMP
         JS    LPMESG    OUTPUT TO LINEPRINTER
          LAE  ERRM1
          STA  DUMMY
          JS   MESSOUT
         JGU   INCR       GO INCREMENT BUFFER POINTER
*
ERMES2    LDA  8,M
          STA  KOUNT
          LAE  ERRM2
          STA  DUMMY
          JS   MESSOUT
         JGU   INCR       GO INCREMENT BUFFER POINTER
*
ERMES3    LDA  7,M
          STA  KOUNT
          LAE  ERRM3
          STA  DUMMY
          JS   MESSOUT
          JGU  CONTINUE
*
* MESSAGE OUTPUT ROUTINE
*
MESSOUT   PTR  MESSORTN
          LDX  4,0,M
          LDB  DUMMY,I
         JS    LPMESG    OUTPUT TO LINEPRINTER
          LAE  DUMMY,I
          ADU  2,M
          STA  DUMMY
          IMP  4,1,M
          ICL  4,KOUNT
          RTA  MESSORTN
          JGU  MESSOUT+4
*
MESSO     PTR  RTN2
          LDA  15,M
          STA  KOUNT
          LAE  MESS1
          STA  DUMMY
          JS   MESSOUT
          RTA  RTN2
*
```

```
* PROGRAM TO READ CARDS AND LIST ON LINEPRINTER
*
* SWITCH  3 INHIBITS LINEPRINTER
*
RRRTN     HEX    0
*
RDPRNT    PTR    RRRTN
NXTCD     JS     READCD    GO READ A CARD
          LDX    1,0,M   INITIALIZE BUFFER INDEX
PRNTCD    LDBH   CARDBUFF,1    GET STORED CHARACTER
          JS     LPMESG    OUTPUT CHARACTER
          IMP    1,1,M   INCREMENT BUFFER INDEX
          ICL    1,80,M    TEST FOR END OF CARD
          JGU    TERMLN    END, TERMINATE LINE ON LINEPRINTER WITH CR
          JGU    PRNTCD    NOT END, PRINT NEXT CHARACTER
TERMLN    LDB    LEFTCR    LOAD B REG WITH CODE FOR CARRIAGE RETURN
          JS     LPMESG    OUTPUT CARRIAGE RETURN TO LINEPRINTER
          RTA    RRRTN   RETURN
*
*
* SUBROUTINE READCD
* READ ONE CARD INTO CARDBUFF
* DATA IS STORED IN HALFWORDS  LEFT JUSTIFIED
*
LEFTCR    HEX    0D000000
CARDRTN   HEX    0
CARDBUFF  BSS    80
*
READCD    PTR    CARDRTN
          LDX    1,0,M   INITIALIZE BUFFER INDEX
CDWT      DIA    23        GET CARDREADER STATUS
          SAM    256,M   CHECK CR READY
          JGU    CDWT   WAIT FOR READY
CDLOOP    DIA    23   READ CARDREADER STATUS
          SAM    4096,M    CHECK FOR CYCLE FINISHED
          JGU    CDU   CYCLE COMPLETE, CONTINUE
          JGU    CDLOOP    NOT FINISHED YET, CHECK AGAIN
*
CDU       DOA    23,C   PICK A CARD
CD2       DIA    23   READ STATUS
          SAM    512,M   CHECK IF DATA RDY
          JGU    CD2   NOT READY, CK AGAIN
          AND    127,M   RDY, KEEP 7 DATA BITS
          SLL    24        LEFT JUSTIFY
          STAH   CARDBUFF,1    STORE CHARACTER
          IMP    1,1,M   INCREMENT BUFFER INDEX
          ICL    1,80,M    TEST FOR END OF CARD
          RTA    CARDRTN    END, EXIT
          JGU    CD2    NOT END, GET NEXT CHARACTER
*
*
*     SUBROUTINE LPMESG            CALL JS LPMESG
*                                  DATA TO BE PRINTED IN B REG LEFT JUSTIFD
*
LPRTN      HEX     0
*
LPMESG      PTR    LPRTN
          _JGW    LP3,3  SKIP LINE PRINTER IF SW 3 SET
*    SEND REMOTE ENABLE
         LDA    2848,M
         DOA    22,C,K
*      CHECK FOR RDY
         DIA    22      READ STATUS
         SRC    2
         JL     *-2     NOT RDY,WAIT
*      SEND CHAR MODE SELECT
         LDA    2880,M
         DOA    22,C,K
*      CHECK FOR NOT BSY
         DIA    22
         SRC    3
         JL    *-2     BSY, WAIT
*     PUT ASCII CHARS OUT TO LP
LP2      LDA    0,M
         SLLD     8      BRING NEXT CHAR TO A
         JN    LP1      NOT ZERO,PROCEED
         JU    BOF      ZERO DATA,CK BOF AND RETURN
LP1      LOR    2048,M     SET SINGLE CHAR MODE
         DOA    22,C,K     OUTPUT CHAR
*      CHECK FOR RDY,BSY,PC BSY
         DIA    22
         SAM    14,M     CK 3 BITS
         JGU    LP2     OK,GET NEXT CHAR
         JGU    *-5     NOT OK, WAIT
*
BOF      DIA    22      CHECK FOR BOTTOM OF FORM
         SRC    5
         JRG    LP3     NO BOF, RETURN
*     BOF FOUND - ISSUE TOF
         LDA    2995,M
         DOA    22,C,K
*      WAIT FOR PC NOT BSY
         DIA    22
         SRC    4
         JL    *-2     BSY, WAIT
LP3      RTA    LPRTN       RETURN
```

```
*
* PUNCH ROUTINE (ASSUMES DATA IN B REGISTER)
*
PUNCHR   PTR  PUNRTN
         LDA  INSTR1  SET UP TRAP
          STA  I14TRP  JMP FOR INT14
          LDA  ENABLP   ENABLE PUNCH
          DOA  16,C,K  ISSUE COMMAND
          LDA  0,M
PCYL      SLLD 8
           JN   PUN    JMP IF NOT 0
          LDA  DISIO  DISABLE ALL.DEVICES
          DOA  16,C,K  ISSUE COMMAND
          DPI          DISABLE PROGRAM INT
          RTA  PUNRTN
*
PUN       DOA  16,K     OUTPUT CHAR
          LDA  INT14
          STA  TEMP
          LDI  INT14
          EPI
* WAIT FOR FINISH LOOP
WAIT3      LDA  TEMP   IS FLG SET
           JN   WAIT3
           JU   PCYL
* TRAP LOOP FOR PUNCH
PUNTRP    LDA  0,M
          STA  TEMP
          LDA  ENABLP  RS PUN INT
          DOA  16,C,K
          RTA  I14RTN
*
*
ERRM1      HEX  202D2D2D  SP---
           HEX  494E5641  INVA
           HEX  4C494420  LID_
           HEX  43484152  CHAR
           HEX  0D0A2020  CRLF__
*
ERRM2      HEX  0D0A2020  CRLF__
           HEX  494E5055  INPU
           HEX  54204441  T_DA
           HEX  54412020  TA__
           HEX  4F555420  OUT_
           HEX  4F462052  OF_R
           HEX  414E4745  ANGE
           HEX  0D0A2020  CRLF__
*
```

```
ERRM3       HEX   0D0A2020   CRLF__
            HEX   434F4E56   CONV
            HEX   20444154   _DAT
            HEX   41204F55   A_OU
            HEX   54204F46   T_OF
            HEX   2052414E   _RAN
            HEX   47450D0A   GECRLF
*
MESS1       HEX   0D0A2020   CRLF__
            HEX   44415441   DATA
            HEX   20434F4D   _COM
            HEX   50524553   PRES
            HEX   53494F4E   SION
            HEX   2050524F   _PRO
            HEX   4752414D   GRAM
            HEX   0D0A2020   CRLF__
            HEX   53574954   SWIT
            HEX   43482023   CH_#
            HEX   30205354   0_ST
            HEX   4F505320   OPS_
            HEX   5052494E   PRIN
            HEX   54204F55   T_OU
            HEX   54200D0A   T_CRLF
*
            END
            END
>
```